Date: 11.01.2021
Author: Vesa Bianca

3D graphic photorealistic scene

Kids Toy Room

Student: Vesa Bianca

Grupa: 30237

MINISTRY OF EDUCATION
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

Date: 11.01.2021
Author: Vesa Bianca

Contents

Date: 11.01.2021
Author: Vesa Bianca

I Prezentarea temei

Scopul acestui proiect este realizarea unei prezentari fotorealiste a unei scene de obiecte 3D, utilizand librariile prezentate la materia Prelucrare Grafica (OpenGL, GLFW, GLM etc.). Totalitatea obiectelor prezente in aceasta scena reprezinta o camera cu jucarii pentru copii.

II Scenariul

Scenariul acestei scene este ilustrarea unei reprezentari a unei camere cu jucarii, care poate fi analizata, si cu care se poate interactiona prin intermediul tastaturii si a mouse-ului.

2.1 Descrierea scenei și a obiectelor

Obiectul de prim-plan al acestei scene este incaperea cu 3 pereti si podea in care sunt pozitionate obiectele. Ea contine o masa, pe care este asezata o lampa de birou, o lampa verticala de podea, precum si un dulap cu rafturi si o usa.

Toate celelalte obiecte sunt independente intre ele. Acestea sunt: un covor asezat pe podea, o sanie de lemn, o racheta si o minge de tenis, o casa de papusi, o papusa de plastic, un caine de plus, un avion telecomandat, o fabrica de dulciuri de jucarie, un ponei de plastic, o bicicleta de lemn, un set de zaruri numerotate, o figurina LEGO, o papusa din material textil. Pe masa am asezat un set de creioane colorate si o ceasca de ciocolata calda, iar pe scaun se afla o vulpe de plus. In aer se afla un balon caruia i-am adaugat functionalitatea de a se translata constant, fara vreo interventie de la utilizator, in jurul unui punct din spatiu, pentru a da efectul de balon cu heliu care pluteste. Pe rafturile dulapului se afla o pusculita in forma de pisica, o minge de fotbal, o macheta a unui hidroavion si o masina de jucarie. Pentru a da simetrie scenei am ales sa o completez cu o fotografie inramanta cu personaje din desene animate, situate pe un perete. Pe peretele opus am atasat alte doua rafturi de lemn, pe care se afla o alta poza inramata, respectiv un set de carti.

Toate obiectele au fost preluate de pe site-urile web Turbosquid [1], Sketchfab [2] si Free3D [3] , iar eventualele imagini necesare pentru maparea texturilor obiectelor au fost preluate de pe site-urile CC0 Textures [4] si Texturise [5].

2.2 Funcționalități

Scena se deschide cu o animatie de prezentare. Camera cu care este vizualizata scena este adusa in centrul incaperii, apoi este rotita 360 de grade pentru a permite o previzualizare de ansamblu a tuturor obiectelor, dupa care este adusa inapoi in pozitia initiala.

Dupa incheierea animatiei de prezentare, aplicatia poate primi comenzi de la tastatura si mouse pentru a interactiona cu obiectele si a naviga prin scena cu ajutorul camerei.De asemenea, scena se poate vizualiza in diferite moduri (Wireframe, Fill si Point) din punct de vedere al trasarii obiectelor, iar din punct de vedere al iluminarii, se poate trece din modul zi in modul noapte si viceversa.

Figura 1. Scena in modul zi

Date: 11.01.2021
Author: Vesa Bianca

Figura 2. Scena in modul noapte

III Detalii de implementare

3.1 Funcții și algortmi

Functiile adaugate scheletului initial al proiectului sunt:

- void getPointLightPos() si void getSpotLightPos(), care se apeleaza in functia principala ce se ocupa cu desenarea intregii scene, si anume renderScene(). Aceste functii realizeaza transmiterea variabilelor pointLightPosEye si spotLightPosEye catre fragment shader la trasarea fiecarui frame de catre placa grafica. Variabilele mentionate anterior reprezinta pozitiile in eye space a surselor luminii punctiforme si de tip spot si sunt necesare in cazul aplicarii unei rotatii sau a unei translatii asupra obiectului camera, fapt care ar duce la modificarea pozitilor exacte alese initial a surselor de lumina in spatiu.

Date: 11.01.2021
Author: Vesa Bianca

```
void getPointLightPos() {

    myBasicShader.useShaderProgram();

    pointLightPos = glm::vec3(-0.919999f, 0.45f, -0.54f);   // floor lamp

    model = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, 0.0f));
    model = glm::rotate(model, glm::radians(angle), glm::vec3(0, 1, 0));
    pointLightPos = glm::vec3(model * glm::vec4(pointLightPos, 1.0f));

    pointLightPosV = glm::vec4(pointLightPos, 1.0f);

    pointLightPosLoc = glGetUniformLocation(myBasicShader.shaderProgram, "pointLightPosEye");
    glm::vec4 ptr = glm::mat4(view) * pointLightPosV;
    glUniform3fv(pointLightPosLoc, 1, glm::value_ptr(glm::vec3(ptr)));

}
```

Figura 3. Functia getPointLightPos

```
void getSpotLightPos() {

    myBasicShader.useShaderProgram();

    spotLightPos = glm::vec3(0.62f, 1.09f, 1.12f);    // desk lamp

    model = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, 0.0f));
    model = glm::rotate(model, glm::radians(angle), glm::vec3(0, 1, 0));
    spotLightPos = glm::vec3(model * glm::vec4(spotLightPos, 1.0f));

    spotLightPosV = glm::vec4(spotLightPos, 1.0f);

    spotLightPosLoc = glGetUniformLocation(myBasicShader.shaderProgram, "spotLightPosEye");
    glm::vec4 ptr = glm::mat4(view) * spotLightPosV;
    glUniform3fv(spotLightPosLoc, 1, glm::value_ptr(glm::vec3(ptr)));

}
```

Figura 4. Functia getSpotLightPos

- renderRoom(shader, showMap), renderMovingPlane(shader, showMap), renderRug(shader, showMap), renderBike(shader, showMap), renderMug(shader, showMap), renderBalloon(shader, showMap), renderRacket(shader, showMap), renderToyPlane(shader, showMap), renderFox(shader, showMap), renderPony(shader, showMap), renderDogToy(shader, showMap), renderSled(shader, showMap), renderTennisBall(shader, showMap), renderDoll(shader, showMap), renderDollHouse(shader, showMap), renderSoccerBall(shader, showMap), renderPonyHouse(shader, showMap), renderCrayons(shader, showMap), renderPaperDoll(shader, showMap), renderCatToy(shader, showMap), renderFigurine(shader, showMap), renderNumberedDice(shader, showMap), renderTruckToy(shader, showMap), renderFirstShelf(shader, showMap), renderSecondShelf(shader, showMap), renderPicture(shader, showMap), renderFrame(shader, showMap) si renderBooks(shader, showMap) sunt functiile care realizeaza operatiile de translatie, rotatie si scalare asupra matricei model, ce se va aplica in interiorul fragment shader-ului asupra fiecarui obiect. Cu ajutorul acestor operatii, am reusit sa pozitionez fiecare obiect la coordonatele dorite in scena si sa le redimensionez corespunzator.

```
void renderRoom(gps::Shader shader, bool showMap) {
    // select active shader program
    shader.useShaderProgram();

    //send model matrix data to shader
    model = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, 0.0f));
    model = glm::rotate(model, glm::radians(angle), glm::vec3(0, 1, 0));
    model = glm::translate(glm::mat4(model), glm::vec3(-0.18f, -0.38f, -0.1f));

    model = glm::scale(model, glm::vec3(0.6f));

    glUniformMatrix4fv(glGetUniformLocation(shader.shaderProgram, "model"), 1, GL_FALSE, glm::value_ptr(model));


    normalMatrix = glm::mat3(glm::inverseTranspose(view * model));

    //send normal matrix data to shader
    if (!showMap)
        glUniformMatrix3fv(normalMatrixLoc, 1, GL_FALSE, glm::value_ptr(normalMatrix));

    // draw model
    room.Draw(shader);
}
```

Figura 5. Exemplu de functie de rendering pentru un obiect, incaperea cu jucarii

- Un alt aspect care merita mentionat este calculul culorii finale in fragment shader. Astfel, pentru a fi posibila schimbarea la modul “noapte” respectiv modul “zi” se tine cont de o variabila de tip uniform trimisa shader-ului. La comanda utilizatorului de la tastatura, variabila respectiva isi va schimba valoarea din vec3(0.0, 0.0, 0.0) in vec3(1.0, 1.0, 1.0), semnaland shaderului sa calculeze culoarea fiecarui fragment, tinand cont doar de lumina punctiforma si de cea de tip spot, fara a adauga si efectul luminii directionale.

```
void main()
{
    if( night.x == 1.0f && night.y == 1.0f && night.x == 1.0f){
    color = computeSpotLight() +  computePointLight();
    }

    if( night.x == 0.0f && night.y == 0.0f && night.x == 0.0f){
    color =  computeDirLight();
    }

    //compute final vertex color
    fColor = vec4(color, 1.0f);
}
```

Figura 6. Calcularea culorii finale a fiecarui pixel in fragment shader

3.2 Modelul grafic

Din punct de vedere al graficii 3D, am utilizat pentru fiecare obiect din scena un fisier de tip obj, impreuna cu fisierul .mtl corespunzator, care are rolul de a mapa texturile pe respectivul obiect. O parte dintre obiecte au fost modificate dupa descarcarea lor, pentru a se potrivi viziunii pe care am construit-o atunci cand am ales ideea de baza a proiectului. Pentru modificarea texturilor am ales sa editez fisierele .mtl intr-un editor text, folosindu-ma de diferite texturi pe care le-am gasit potrivite, iar pentru eventualele modificari la nivelul obiectelor am utilizat programul software Blender [6].

Date: 11.01.2021
Author: Vesa Bianca

3.3 Structuri de date

Structurile de date de baza utilizate in implementarea acestui proiect sunt cele puse la dispozitie in scheletul proiectului, care au fost discutate pe parcursul semestrului la material Prelucrare Grafica. Camera (obiectul care permite vizualizarea scenei), Model3D (care reprezinta abstractizarea fiecarui obiect necesar in dezvoltarea scenei), Shader (entitatea de baza care trimite comenzi placii grafice, pentru a obtine imaginea finala a fiecarui pixel din fiecare frame), SkyBox (clasa care contureaza imaginea de fundal a scenei pentru a creste fotorealismul), precum si diferite alte structuri si tipuri de date puse la dispozitie de librariile OpenGL, GLFW si GLM, cum ar fi glm::mat4, glm::vec3, GLuint, GLfloat.

3.4 Ierarhia de clase

Ierarhia de clase a proiectului este structurata astfel:

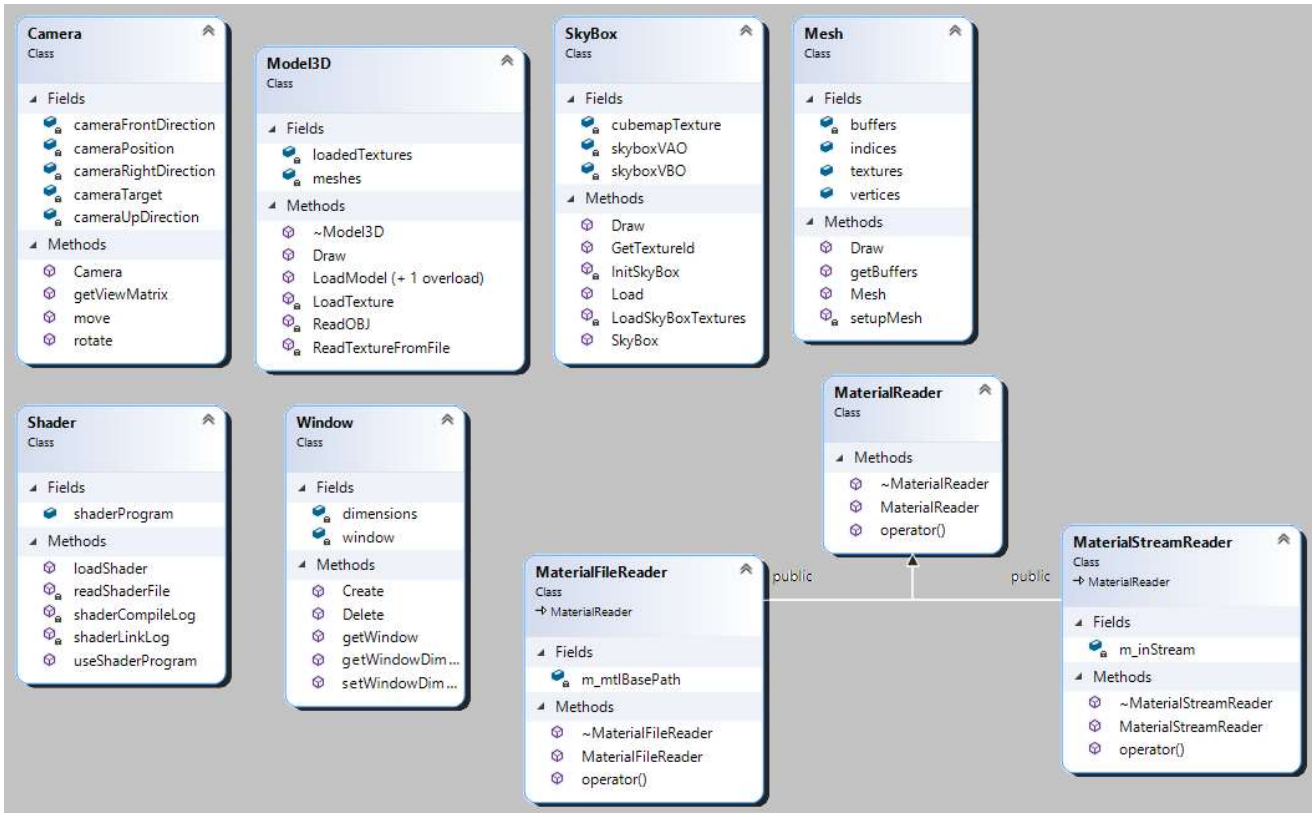

Figura 7. Ierarhia de clase

Date: 11.01.2021
Author: Vesa Bianca

IV Prezentarea interfeței grafice utilizator / manual de utilizare

Aplicatia poate primi comenzi de la utilizator prin intermediul mouse-ului si a tastaturii.

Cu ajutorul mouse-ului, prin apasarea butonului din stanga, se roteste intreaga incapere, impreuna cu obiectele sale, in sensul acelor de ceas, iar la apasarea butonului din dreapta, se executa rotatia in sensul invers acelor de ceas. Comenzile primite de la tastatura sunt urmatoarele:

- Tasta Q executa aceeasi rotatie in sensul acelor de ceas precum butonul stanga al mouse-ului
- Tasta E executa aceeasi rotatie in sensul invers acelor de ceas precum butonul dreapta al mouse-ului
- Tasta W translateaza in fata camera cu care se vizualizeaza scena
- Tasta S translateaza in spate camera cu care se vizualizeaza scena
- Tasta A translateaza in stanga camera cu care se vizualizeaza scena
- Tasta D translateaza in dreapta camera cu care se vizualizeaza scena
- Tasta T translateaza in sus camera cu care se vizualizeaza scena
- Tasta G translateaza in jos camera cu care se vizualizeaza scena
- Tasta X dezactiveaza lumina directionala si activeaza lumina punciforma si cea de tip spot pentru modul “noapte”, respectiv la a doua apasare, activeaza la loc lumina directionala si le dezactiveaza pe cele de tip punctiforma si spot pentru modul “zi”
- Tasta P activeaza trasarea scenei in modul Point
- Tasta L activeaza trasarea scenei in modul Wireframe
- Tasta F activeaza trasarea scenei in modul Fill, adica modul in care se traseaza initial
- La apasarea tastei Z, avionul telecomandat asezat pe podea va incepe sa execute anumite translatii si rotatii prin scena pentru a da impresia de zbor

Date: 11.01.2021
Author: Vesa Bianca

V Concluzii și dezvoltări ulterioare

Acest proiect a jucat un rol important in ceea ce priveste punerea in practica a informatiilor invatate in decursul acestui semestru la materia Prelucrare Grafica, deoarece imbina intr-un mod echilibrat concepte din fiecare sedinta de laborator. Astfel, implementarea acestuia s-a desfasurat pas cu pas, inradacinand mai bine notiunile invatate.

Ca dezvoltari ulterioare se merita mentionate functionalitati precum miscarea camerei in mai multe moduri cu ajutorul mouse-ului, si nu doar de tip rotatie la apasarea butoanelor. Posibilitatea de a afisa umbre generate si de catre sursele de lumina punctiforma si de tip spot este o posibila imbunatatire a aplicatiei. De asemenea, pentru a creste fotorealismul scenei, se pot adauga animatii diverse anumitor obiecte sau a anumitor componente ale acestora.

VI Referinte

[1] "Turbosquid," [Online]. Available: https://www.turbosquid.com/.

[2] "Sketchfab," [Online]. Available: https://sketchfab.com/feed.

[3] "Free3D," [Online]. Available: https://free3d.com/.

[4] "CC0 Textures," [Online]. Available: https://cc0textures.com/.

[5] "Texturise," [Online]. Available: http://www.texturise.club/.

[6] "Blender," [Online]. Available: https://www.blender.org/.